An AI Approach for Scheduling Space-Station Payloads at Kennedy Space Center

by

D. Castillo, D. Ihrie
HARRIS Corporation
Melbourne, FL 32935

M. McDaniel
MDAC-KSC
KSC, FL 32899

R. Tilley
NASA/SS-OCO
KSC, FL 32899

ABSTRACT

Payload Processing for Space-Station Operations, including mission manifesting and its effect on KSC Ground Resource Allocation, represents a class of ill-structured, complex scheduling problems which are often unsuitable for applying optimization algorithms. The situation has inspired the development of AI-based planning and scheduling systems specifically designed for Payload Processing activities. This paper examines the application of an AI-based system, called PHITS, to integrated payload scheduling and its effect on Ground Resource Allocation at KSC.

Unique to the PHITS approach is the process by which schedule generation occurs. Experiments are represented in terms of objects which are semantically related based on mission goals. Unlike conventional scheduling systems, task flows are only defined for individual objects. Integrated schedules are generated by evaluating object, attribute, value (OAV) triplets for experiments considered candidates for flight. OAV triplets contain user-defined constraints on object interaction. A goal directed simulation subsystem examines the schedule and performs conflict resolution as needed to achieve the on-orbit requirement goal.

INTRODUCTION

Part of Kennedy Space Center's (KSC) responsibility is the prelaunch preparation and integration of experiments for transport to Space Station and the deintegration and processing of returned hardware. As an example, consider an experiment which operates in a pressurized laboratory module. The experiment is first received at the launch site where it is inspected and integrated into a laboratory equipment rack. A high fidelity simulator then verifies the module-to-rack interfaces prior to installing the rack in a logistics module. Finally, the logistics module is installed in the orbiter for transport to the Space Station. Each payload generates unique and complex demands which require a large array of resources including facilities, equipment, materials, personnel skills and training. Furthermore, the process becomes even more complex when processing multiple shuttle flights in parallel. As a result of the relatively fixed level of available resources and

highly structured set of schedule constraints imposed by the shuttle launch and landing, planning and scheduling of processing activities associated with new flights represents a nontrivial task. Efficient planning and management of this process is a key element in maximizing the effective use of the Space Station while minimizing the cost.

The above situation has directed researchers toward AI-based scheduling systems designed to operate in the domain of resource-constrained scheduling. This effort has produced systems such as MAESTRO [2], PLANNET [8], MARS [9], PEGASUS [4] and others. A commonality among many of these systems is the methodology used in generating conflict-free schedules. In many cases, a schedule is generated using traditional CPM routines, followed by heuristic methods that attempt to produce a conflict free schedule. More recently, researchers have investigated the effects of temporal reasoning applied to resource-constrained scheduling [1] [11] in hopes of automating deductions about time.

The Payload Handling Inventory Tracking System (PHITS), developed by Harris Corporation [5], deviates from the above philosophical methodologies in that it provides a modeling environment that couples scheduling, simulation and AI technologies in one unique modeling environment. Bruno et al. [3] share this philosophy and have applied it to the domain of Flexible Manufacturing Systems (FMS). However, the distinction between PHITS and Bruno's FMS system is the way schedule generation and simulation are performed. This paper examines the application of PHITS to integrated payload scheduling and the effects this has on Ground Resource Management giving particular attention to the usability, scheduling and simulation aspects of PHITS. The reader is referred to Ihrie et al. [6] for an overview of the technologies used in PHITS.

STORAGE STUDY OBJECTIVES

MDAC-KSC was tasked to identify Space Station payload storage policy alternatives at KSC. This required a forecast of storage requirements in relationship to experiment types, sizes and numbers that flowed through the processing facilities at KSC. It was determined that a software tool capable of tracking resources and forecasting their requirements in a very dynamic environment would greatly assist MDAC-KSC in accomplishing the objectives of the study. In addition, a "what if" feature for performing sensitivity analysis on the primary variables defined in the study would provide the flexibility for examining competing scenarios. MDAC in cooperation with NASA-KSC agreed to use the PHITS system for supporting their study efforts.

Having defined the manifest as the primary variable for determining storage study policies, it was recognized that the ability to generate manifests was necessary since manifests were generally unavailable for most flights considered in the study. PHITS possessed the capability to generate manifests based on

experiments from the Civil Needs Database and their associated constraints. Flights from 1994 to 2002 including 127 experiments were considered. Based on the results of the manifest, PHITS produced a payload schedule defining the timeline for all tasks and resources. The schedule was then simulated to determine storage requirements and resolve any resource conflicts that occurred. The following sections describe this process.

BUILDING THE STORAGE STUDY MODEL

Developing the Storage Study Model required the user to define the objects relevant to the study, such as experiments, log-modules etc. PHITS provides a powerful user interface for simplifying the process of identifying and defining objects. The Genealogy Editor was utilized to identify each object as a class or instance and graphically portray all parent-child relationships. Figure 1 illustrates the Genealogy Editor.

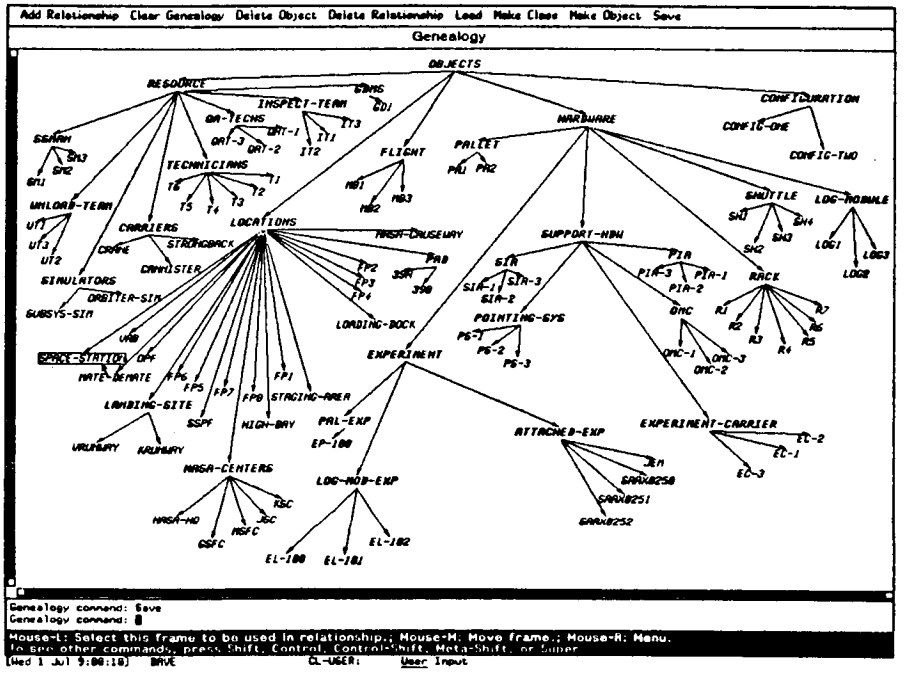

Figure-1: PHITS Genealogy Editor

Object task flow scripts were then defined for each object. Task flows were either inherited from a class of Experiment, or uniquely created for the specific experiment being defined. Task-based resources and storage types were identified during this process. OAV requirements were also instantiated at this time. OAV triplets represent constraints one object imposes on other objects. In the Storage Study, for example, an experiment object that affects available shuttle mass is represented by the following OAV triplet:

```
((shuttle mass 2345))
```

PHITS uses this property list to determine if the experiment is capable of being attached to the flight. The scheduling component utilizes OAV triplets for developing the entire task network for a given mission.

Once all objects were identified and defined in terms of their task network flows, the Attribute Editor was accessed for defining object attributes. For example, object MB-3 contains an attribute called "launch-date" with a facet of "is" and a value of "July 15, 1994." For the Storage Study, each experiment contained an attribute called "sq-ft" which represented the square-foot dimensions of the experiment. Furthermore, each experiment class contained an algebraic expression attribute utilized to compute storage requirements based on the sq-ft attribute of each experiment.

PHITS provides a Structure Editor for attaching experiments to a given flight. Attaching experiments in this manner guarantees the experiment will be manifested during payload scheduling. This is a useful feature when a manifest has been previously set by NASA or when analyzing different manifests.

PAYLOAD SCHEDULING

Payload scheduling in PHITS consists of connecting individual object definitions into a single integrated payload flow that satisfies all connector constraints. Figure 2 illustrates the task flow script for individual objects. From Figure 2 the object shuttle is defined by more than one set of task flows. Each flow may contain one or more start and end nodes. End nodes contain information corresponding to a connecting set of tasks on another object.

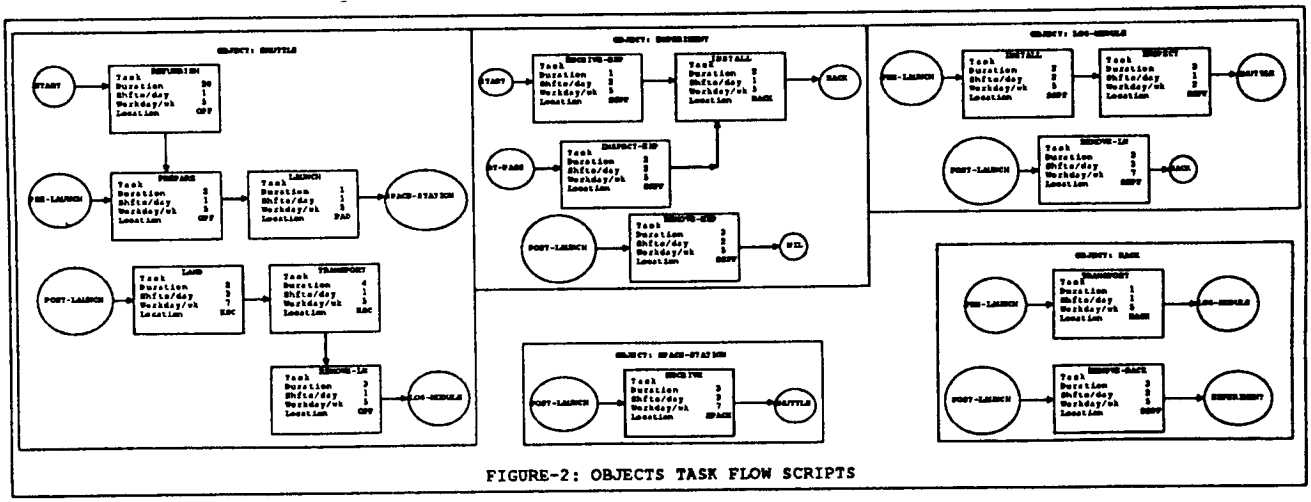

FIGURE-2: OBJECTS TASK FLOW SCRIPTS

The scheduler operates on a user-selected set of missions and experiments. A mission is considered an object with a launch date and possibly containing a set of experiments. Additional experiments are left unattached and may be considered candidates for flight. For each manifested and unmanifested experiment, the

scheduler first constructs a complete stand-alone task flow by following connections between object task flows. OAV triplets are not considered during this forward pass. Figure 3 represents this process:

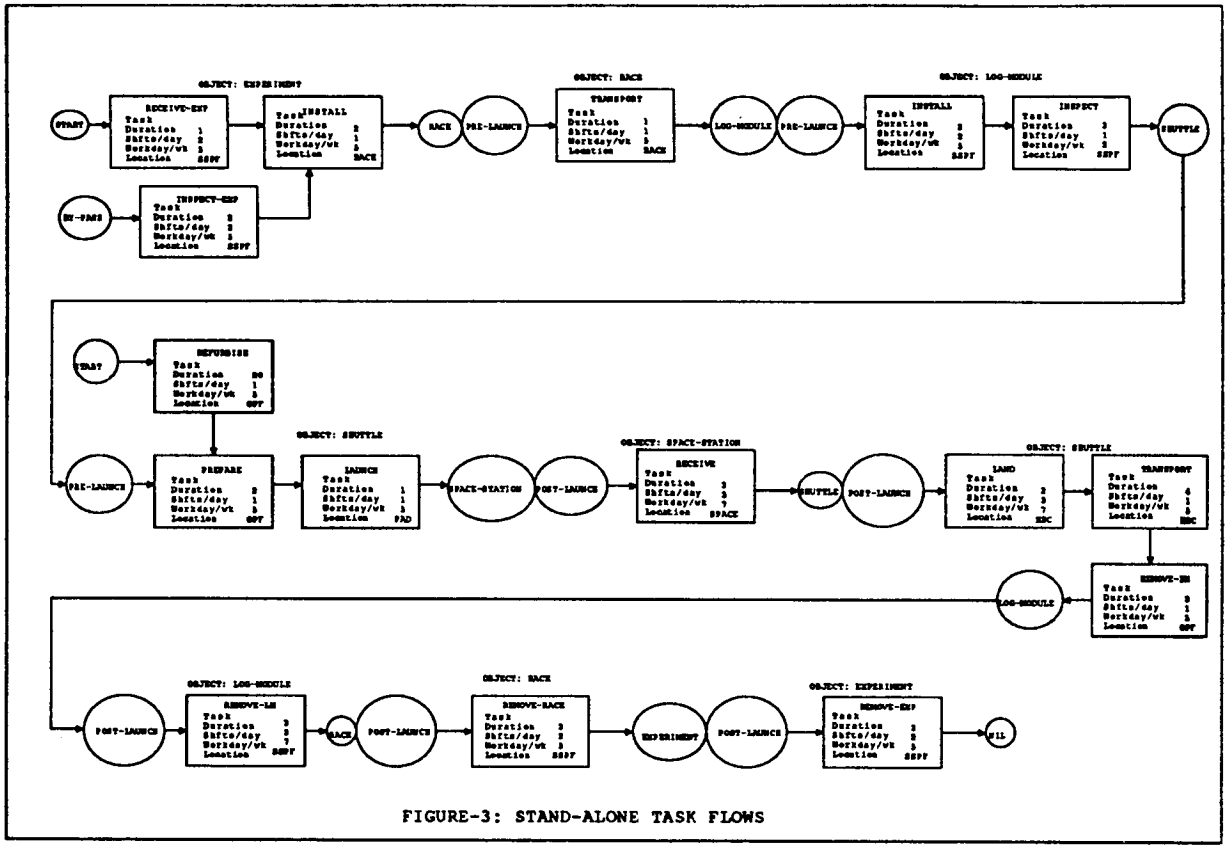

FIGURE-3: STAND-ALONE TASK FLOWS

At this point, the scheduler must eliminate duplicate tasks from experiment flows attached to a single mission, manifest the unmanifested experiments, replace object classes with specific objects and assign task dates to meet launch dates. The key ingredient here is ensuring that all mission constraints are met. Constraints are represented by the OAV triplets mentioned previously. At each point in the flow where multiple objects are integrated, the scheduler attaches as many objects as possible without violating the OAV constraints. Manifested experiments are attached to a mission first, followed by unmanifested experiments wherever possible, based on a user-selected priority. The final output is a fully integrated task flow for each mission, such as that shown in Figure 4. Using this approach, the scheduler guarantees each mission is internally conflict free and satisfies all object constraints.

A CPM routine is then applied to the overall network to instantiate the Early-Start, Late-Start, Early-Finish and Late-Finish dates on the network. The critical path is defined as the task path where Early-Start is equal to Late-Start and Early-Finish is equal to Late-Finish.

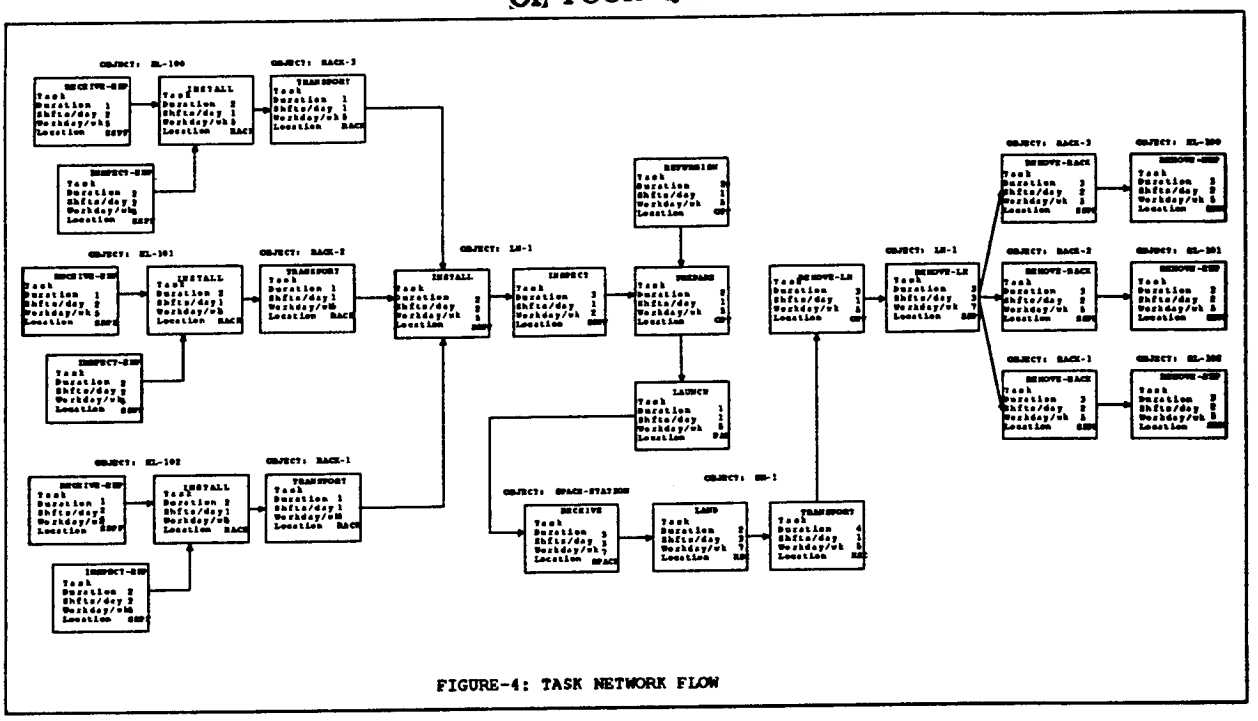

FIGURE-4: TASK NETWORK FLOW

As mentioned above, the scheduler instantiates the early-start late-finish range for each task. This range dictates the permissible time interval of each task. If the range is violated, then the on-orbit requirement date is not guaranteed, resulting in the possibility of a slipped schedule. One of the designated tasks of the simulation subsystem is to ensure schedules are maintained. Figure 5 illustrates the early-start late-finish range for a task.

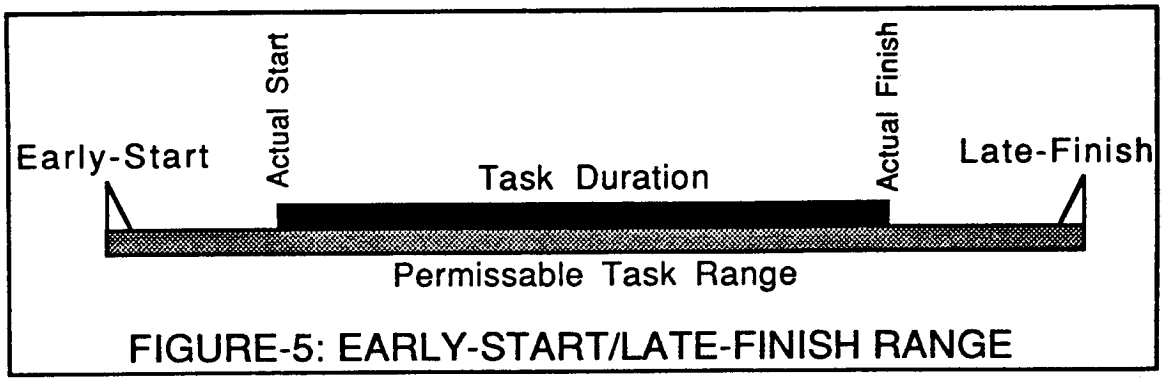

FIGURE-5: EARLY-START/LATE-FINISH RANGE

SIMULATION

Simulation in PHITS represents a complex process of tracking objects, resolving resource conflicts and dynamically re-adjusting in order to achieve the on-orbit requirement goal. An event calendar is utilized as the central control structure that communicates with other objects by message passing in much

the same way as KBS [10] and ROSS [7]. Tasks are examined by the event calendar to determine if a violation in the early-start late-finish range is imminent. If a violation is detected, the task object is given to the Meta-level Resource Manager to determine if a local resolution is feasible. Local resolution implies conflict resolution is applied to the task in question without disrupting the processing of other tasks.

Local resolution is an attempt to quickly resolve resource conflicts by applying a small set of heuristic information on a local scale. PHITS utilizes a Pre-processing facility as a vehicle for performing local resolution. If a task range violation is detected by the event-calendar or Resource Manager, then Pre-processing is initiated. Figure 6 represents a conceptual illustration of a task range violation. Notice the Late-Start milestone has been exceeded, therefore, local resolution will attempt to shorten the task duration so that the task does not violate its Late-Finish range. Pre-processing applies very simple heuristics such as overtime, increased resources where applicable, etc. to satisfy the task's range constraints. If this effort is unsuccessful, then global resolution is initiated. Global resolution is defined as the process where resolving a conflict for one task affects another task's processing. The current version of PHITS contains the architecture for supporting global resolution, however, a significant amount of knowledge engineering is needed before full scale implementation can occur. Although lack of global resolution did not adversely effect the outcome of the Storage Study, it was recognized that other candidate studies would probably require this capability.

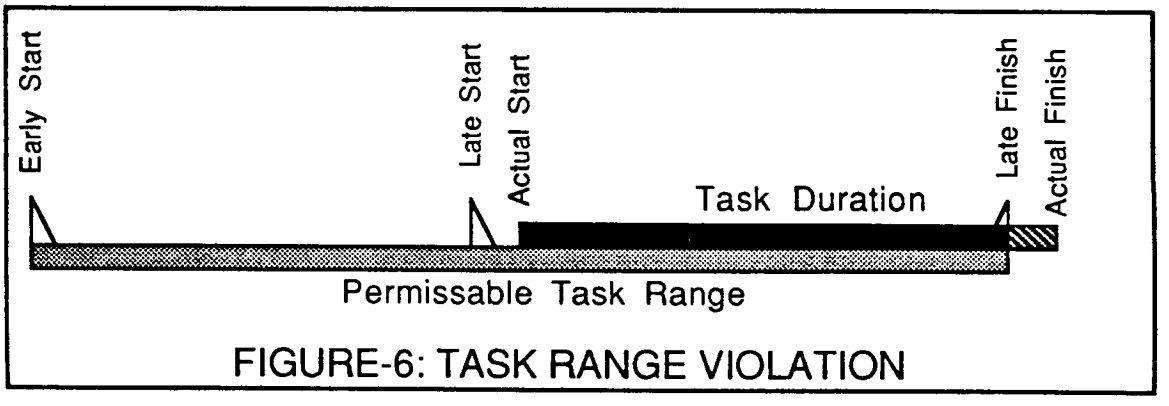

FIGURE-6: TASK RANGE VIOLATION

Due to the object-oriented nature of PHITS, storage requirements were easily generated. Attribute expressions containing algebraic equations and sq-ft values were evaluated at simulation time. Storage requirements based on experiment and storage types were collected. Cumulative storage requirements were reflected graphically as temporal representations which proved useful for comparing multiple manifest scenarios. Gantt charts were utilized for displaying the overall processing schedule. Figures 7 and 8 illustrate these features.

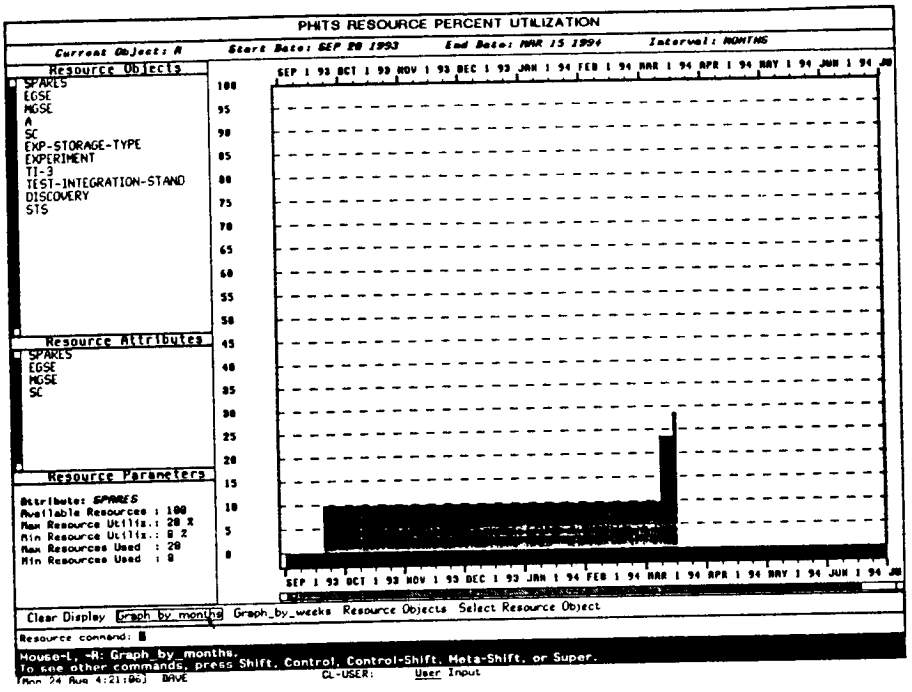

FIGURE-7: RESOURCE UTILIZATION GRAPH

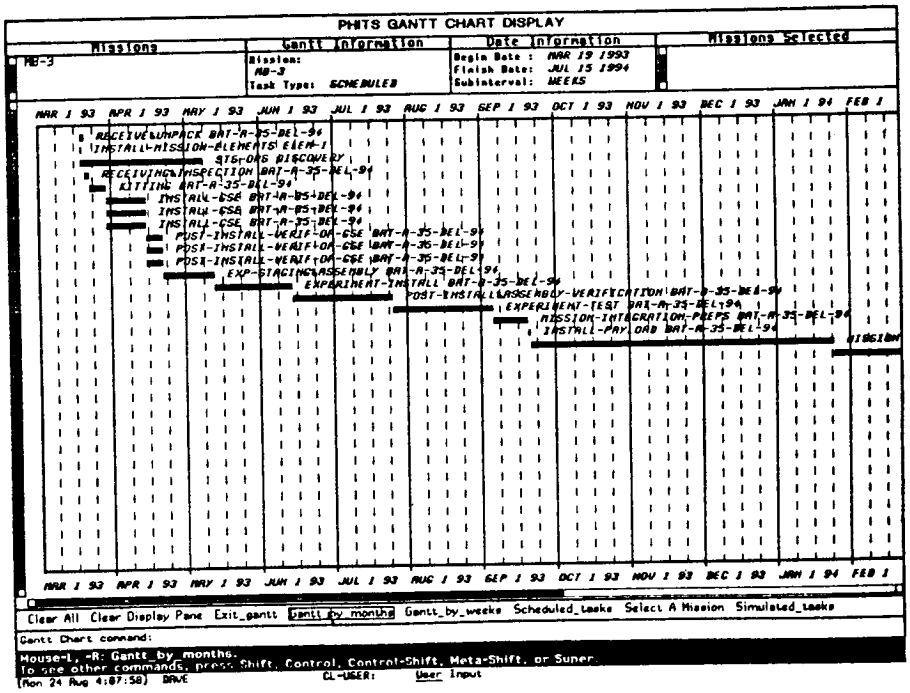

FIGURE-8: GANTT CHART OF SCHEDULE

Although not utilized during the Storage Study analysis, PHITS features an animation component which demonstrates a pictorial view of the simulation process. Object icon attributes are updated dynamically to represent the discrete changes in the simulation status. This proves valuable when an analyst is concerned with tracking specific objects throughout the simulation, or discovering potential bottlenecks of a process.

FUTURE RESEARCH

Future research efforts for PHITS will focus on the Resource Manager and global resolution. Temporal relations will be examined in an effort to better understand and manipulate conflicts on the global scale. Other directions include porting the technology from a Symbolics environment to an 80386 environment. Incorporating intelligence features into the manifesting capability of PHITS also has potential for future research. Finally, the technology contained in PHITS will be investigated for modeling other problem domains within the Space Station Program.

CONCLUSION

PHITS is a prototype modeling tool capable of addressing many Space Station related concerns. The system's object-oriented design approach coupled with a powerful user interface provide the user with capabilities to easily define and model many applications. PHITS differs from many AI-based systems in that it couples scheduling and goal-directed simulation to ensure on-orbit requirement dates are satisfied.

ACKNOWLEDGMENT

This work would have not been possible without the patience and support of the people at MDAC-KSC, particularly John Weber and David Dixon. The authors express their appreciation to these individuals.

REFERENCES

1. Allen, J.F., Koomen J.A., "Planning using a Temporal World Model", Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983, pp. 741-747.

2. Britt, D., Geoffroy, A., Gohring, J., "A Scheduling And Resource Management System For Space Applications", Proceedings on Artificial Intelligence for Space Applications, Huntsville, AL, (Nov.) 1986, pp. 303-310.

3. Bruno, G., Antonio, E, Laface, P., "A Rule-Based System to Schedule Production", COMPUTER, (Jul.) 1986, pp. 32-39.

4. Harris Corporation, "PEGASUS Software User's Manual", 1986.

5. Harris Corporation, "PHITS Phase 3 Quarterly Report", 1987.

6. Ihrie, D., Castillo, D., Kovarik, V., Tilley, R., "PHITS: An Intelligent Modeling Environment for Space Station Operations at Kennedy Space Center", Proceedings Southwest Conference on Simulation and M Huntsville, AL, (Oct.) 1987.

7. McArthur, D., Klahr, P., The ROSS Language Manual, N-1854-AF, The Rand Corporation, 1982.

8. McDonnell Douglas Astronautics Company, The Plannet Expert System: Version 1.1 User's Manual, Project Report to NASA, 1985.

9. McDonnell Douglas Astronautics Company, Mixed Architecture For Reactive Scheduling, contract NAS8-32350, KSC-1, 1986.

10. Reddy, Y.V.Ramana, Fox, M., Hussain, N., McRoberts, M., "The Knowledge-Based Simulation System", IEEE Software, (Mar.) 1986, pp. 26-37.

11. Vere, S., "Planning in Time: Windows and Durations for Activities and Goals", Technical Report, Jet Propulsion Laboratory 1981.